DOI:
10.1007/978-3-662-49529-2_25

Document Version
Peer reviewed version

Linear-Time Sequence Comparison Using Minimal Absent Words & Applications

Maxime Crochemore[1], Gabriele Fici[2], Robert Mercaş[1,3], and Solon P. Pissis[1]

[1] Department of Informatics, King's College London, UK
[2] Dipartimento di Matematica e Informatica, Università di Palermo, Italy
[3] Department of Computer Science, Kiel University, Germany
maxime.crochemore@kcl.ac.uk, gabriele.fici@unipa.it,
rgm@informatik.uni-kiel.de, solon.pissis@kcl.ac.uk

Abstract. Sequence comparison is a prerequisite to virtually all comparative genomic analyses. It is often realized by sequence alignment techniques, which are computationally expensive. This has led to increased research into alignment-free techniques, which are based on measures referring to the composition of sequences in terms of their constituent patterns. These measures, such as q-gram distance, are usually computed in time linear with respect to the length of the sequences. In this article, we focus on the complementary idea: how two sequences can be efficiently compared based on information that does not occur in the sequences. A word is an *absent word* of some sequence if it does not occur in the sequence. An absent word is *minimal* if all its proper factors occur in the sequence. Here we present the first linear-time and linear-space algorithm to compare two sequences by considering *all* their minimal absent words. In the process, we present results of combinatorial interest, and also extend the proposed techniques to compare circular sequences.

Keywords: algorithms on strings, sequence comparison, alignment-free comparison, absent words, forbidden words, circular words.

1 Introduction

Sequence comparison is an important step in many basic tasks in bioinformatics, from phylogenies reconstruction to genomes assembly. It is often realized by sequence alignment techniques, which are computationally expensive, requiring quadratic time in the length of the sequences. This has led to increased research into *alignment-free* techniques. Hence standard notions for sequence comparison are gradually being complemented and in some cases replaced by alternative ones [10]. One such notion is based on comparing the words that are absent in each sequence [1]. A word is an *absent word* (or a forbidden word) of some sequence if it does not occur in the sequence. Absent words represent a type of *negative information*: information about what does not occur in the sequence.

Given a sequence of length n, the number of absent words of length at most n is exponential in n. However, the number of certain classes of absent words is only linear in n. This is the case for *minimal absent words*, that is, absent words in the

sequence whose all proper factors occur in the sequence [5]. An upper bound on the number of minimal absent words is known to be $\mathcal{O}(\sigma n)$ [9,23], where σ is the size of the alphabet Σ. Hence it may be possible to compare sequences in time proportional to their lengths, for a fixed-sized alphabet, instead of proportional to the product of their lengths. In what follows, we consider sequences on a *fixed-sized alphabet* since the most commonly studied alphabet is $\Sigma = \{\mathtt{A},\mathtt{C},\mathtt{G},\mathtt{T}\}$.

An $\mathcal{O}(n)$-time and $\mathcal{O}(n)$-space algorithm for computing all minimal absent words on a fixed-sized alphabet based on the construction of suffix automata was presented in [9]. The computation of minimal absent words based on the construction of suffix arrays was considered in [28]; although this algorithm has a linear-time performance in practice, the worst-case time complexity is $\mathcal{O}(n^2)$. New $\mathcal{O}(n)$-time and $\mathcal{O}(n)$-space suffix-array-based algorithms were presented in [15,2,3] to bridge this unpleasant gap. An implementation of the algorithm presented in [2] is currently, and to the best of our knowledge, the fastest available for the computation of minimal absent words. A more space-efficient solution to compute all minimal absent words in time $\mathcal{O}(n)$ was also presented in [6].

In this article, we consider the problem of comparing two sequences x and y of respective lengths m and n, using their sets of minimal absent words. In [7], Chairungsee and Crochemore introduced a measure of similarity between two sequences based on the notion of minimal absent words. They made use of a length-weighted index to provide a measure of similarity between two sequences, using sample sets of their minimal absent words, by considering the length of each member in the symmetric difference of these sample sets. This measure can be trivially computed in time and space $\mathcal{O}(m+n)$ provided that these sample sets contain minimal absent words of some bounded length ℓ. For unbounded length, the same measure can be trivially computed in time $\mathcal{O}(m^2+n^2)$: for a given sequence, the cumulative length of all its minimal absent words can grow *quadratically* with respect to the length of the sequence.

The same problem can be considered for two *circular* sequences. The measure of similarity of Chairungsee and Crochemore can be used in this setting provided that one extends the definition of minimal absent words to circular sequences. In Section 4, we give a definition of minimal absent words for a circular sequence from the Formal Language Theory point of view. We believe that this definition may also be of interest from the point of view of Symbolic Dynamics, which is the original context in which minimal absent words have been introduced [5].

Our Contribution. Here we make the following threefold contribution:

a) We present an $\mathcal{O}(m+n)$-time and $\mathcal{O}(m+n)$-space algorithm to compute the similarity measure introduced by Chairungsee and Crochemore by considering *all* minimal absent words of two sequences x and y of lengths m and n, respectively; thereby showing that it is indeed possible to compare two sequences in time proportional to their lengths (Section 3).

b) We show how this algorithm can be applied to compute this similarity measure for two circular sequences x and y of lengths m and n, respectively, in the same time and space complexity as a result of the extension of the definition of minimal absent words to circular sequences (Section 4).

c) We provide an open-source code implementation of our algorithms and investigate potential applications of our theoretical findings (Section 5).

2 Definitions and Notation

We begin with basic definitions and notation. Let $y = y[0]y[1]\,..\,y[n-1]$ be a *word* of *length* $n = |y|$ over a finite ordered *alphabet* Σ of size $\sigma = |\Sigma| = \mathcal{O}(1)$. For two positions i and j on y, we denote by $y[i\,..\,j] = y[i]\,..\,y[j]$ the *factor* (sometimes called *substring*) of y that starts at position i and ends at position j (it is empty if $j < i$), and by ε the *empty word*, word of length 0. We recall that a prefix of y is a factor that starts at position 0 ($y[0\,..\,j]$) and a suffix is a factor that ends at position $n-1$ ($y[i\,..\,n-1]$), and that a factor of y is a *proper* factor if it is not y itself. The set of all the factors of the word y is denoted by $\mathcal{F}_y$.

Let x be a word of length $0 < m \leq n$. We say that there exists an *occurrence* of x in y, or, more simply, that x *occurs in* y, when x is a factor of y. Every occurrence of x can be characterised by a starting position in y. Thus we say that x occurs at the *starting position* i in y when $x = y[i\,..\,i+m-1]$. Opposingly, we say that the word x is an *absent word* of y if it does not occur in y. The absent word x of y is *minimal* if and only if all its proper factors occur in y. The set of all minimal absent words for a word y is denoted by $\mathcal{M}_y$. For example, if $y = abaab$, then $\mathcal{M}_y = \{aaa, aaba, bab, bb\}$. In general, if we suppose that all the letters of the alphabet appear in y of length n, the length of a minimal absent word of y lies between 2 and $n+1$. It is equal to $n+1$ if and only if y is the catenation of n copies of the same letter. So, if y contains occurrences of at least two different letters, the length of a minimal absent word for y is bounded from above by n.

A *language* over the alphabet Σ is a set of finite words over Σ. A language is *regular* if it is recognized by a finite state automaton. A language is *factorial* if it contains all the factors of its words. A language is *antifactorial* if no word in the language is a proper factor of another word in the language. Given a word x, the language *generated* by x is the language $x^* = \{x^k \mid k \geq 0\} = \{\varepsilon, x, xx, xxx, \ldots\}$. The *factorial closure* of a language L is the language $\mathcal{F}_L = \{\mathcal{F}_y \mid y \in L\}$. Given a factorial language L, one can define the (antifactorial) language of minimal absent words for L as $\mathcal{M}_L = \{aub \mid aub \notin L, au, ub \in L\}$. Notice that $\mathcal{M}_L$ is not the same language as the union of $\mathcal{M}_x$ for $x \in L$.

We denote by SA the *suffix array* of y of length n, that is, an integer array of size n storing the starting positions of all (lexicographically) sorted suffixes of y, i.e. for all $1 \leq r < n$ we have $y[\mathsf{SA}[r-1]\,..\,n-1] < y[\mathsf{SA}[r]\,..\,n-1]$ [22]. Let $\mathsf{lcp}(r, s)$ denote the length of the longest common prefix between $y[\mathsf{SA}[r]\,..\,n-1]$ and $y[\mathsf{SA}[s]\,..\,n-1]$ for all positions r, s on y, and 0 otherwise. We denote by LCP the *longest common prefix* array of y defined by $\mathsf{LCP}[r] = \mathsf{lcp}(r-1, r)$ for all $1 \leq r < n$, and $\mathsf{LCP}[0] = 0$. The inverse iSA of the array SA is defined by $\mathsf{iSA}[\mathsf{SA}[r]] = r$, for all $0 \leq r < n$. It is known that SA [25], iSA, and LCP [12] of a word of length n can be computed in time and space $\mathcal{O}(n)$.

In what follows, as already proposed in [2], for every word y, the set of minimal words associated with y, denoted by $\mathcal{M}_y$, is represented as a set of tuples $\langle a, i, j\rangle$, where the corresponding minimal absent word x of y is defined by $x[0] = a$, $a \in \Sigma$, and $x[1\,..\,m-1] = y[i\,..\,j]$, where $j - i + 1 = m \geq 2$. It is known that if $|y| = n$ and $|\Sigma| = \sigma$, then $|\mathcal{M}_y| \leq \sigma n$ [23].

In [7], Chairungsee and Crochemore introduced a measure of similarity between two words x and y based on the notion of minimal absent words. Let $\mathcal{M}_x^\ell$ (resp. $\mathcal{M}_y^\ell$) denote the set of minimal absent words of length at most ℓ of x (resp. y). The authors made use of a length-weighted index to provide a measure of the similarity between x and y, using their sample sets $\mathcal{M}_x^\ell$ and $\mathcal{M}_y^\ell$, by considering the length of each member in the symmetric difference $(\mathcal{M}_x^\ell \triangle \mathcal{M}_y^\ell)$ of the sample sets. For sample sets $\mathcal{M}_x^\ell$ and $\mathcal{M}_y^\ell$, they defined this index to be

$$\mathsf{LW}(\mathcal{M}_x^\ell, \mathcal{M}_y^\ell) = \sum_{w \in \mathcal{M}_x^\ell \triangle \mathcal{M}_y^\ell} \frac{1}{|w|^2}.$$

This work considers the following generalized version of the same problem.

MAW-SequenceComparison
Input: a word x of length m and a word y of length n
Output: $\mathsf{LW}(\mathcal{M}_x, \mathcal{M}_y)$, where $\mathcal{M}_x$ and $\mathcal{M}_y$ denote the sets of minimal absent words of x and y, respectively.

We also consider the aforementioned problem for two circular words. A circular word of length m can be viewed as a traditional linear word which has the left- and right-most letters wrapped around and stuck together in some way. Under this notion, the same circular word can be seen as m different linear words, which would all be considered equivalent. More formally, given a word x of length m, we denote by $x^{\langle i\rangle} = x[i\,..\,m-1]x[0\,..\,i-1]$, $0 \leq i < m$, the i-th *rotation* of x, where $x^{\langle 0\rangle} = x$. Given two words x and y, we define $x \sim y$ if and only if there exist i, $0 \leq i < |x|$, such that $y = x^{\langle i\rangle}$. A *circular word* $\tilde{x}$ is a conjugacy class of the equivalence relation $\sim$. Given a circular word $\tilde{x}$, any (linear) word x in the equivalence class $\tilde{x}$ is called a *linearization* of the circular word $\tilde{x}$. Conversely, given a linear word x, we say that $\tilde{x}$ is a *circularization* of x if and only if x is a linearization of $\tilde{x}$. The set $\mathcal{F}_{\tilde{x}}$ of factors of the circular word $\tilde{x}$ is equal to the set $\mathcal{F}_{xx} \cap \Sigma^{\leq |x|}$ of factors of xx whose length is at most $|x|$, where x is any linearization of $\tilde{x}$.

Note that if $x^{\langle i\rangle}$ and $x^{\langle j\rangle}$ are two rotations of the same word, then the factorial languages $\mathcal{F}_{(x^{\langle i\rangle})^*}$ and $\mathcal{F}_{(x^{\langle j\rangle})^*}$ coincide, so one can unambiguously define the (infinite) language $\mathcal{F}_{\tilde{x}^*}$ as the language $\mathcal{F}_{x^*}$, where x is any linearization of $\tilde{x}$.

In Section 4, we give the definition of the set $\mathcal{M}_{\tilde{x}}$ of minimal absent words for a circular word $\tilde{x}$. We will prove that the following problem can be solved with the same time and space complexity as its counterpart in the linear case.

MAW-CircularSequenceComparison
Input: a word x of length m and a word y of length n
Output: $\mathsf{LW}(\mathcal{M}_{\tilde{x}}, \mathcal{M}_{\tilde{y}})$, where $\mathcal{M}_{\tilde{x}}$ and $\mathcal{M}_{\tilde{y}}$ denote the sets of minimal absent words of the circularizations $\tilde{x}$ of x and $\tilde{y}$ of y, respectively.

3 Sequence Comparison

The goal of this section is to provide the first linear-time and linear-space algorithm for computing the similarity measure (see Section 2) between two words defined over a fixed-sized alphabet. To this end, we consider two words x and y of lengths m and n, respectively, and their associated sets of minimal absent words, $\mathcal{M}_x$ and $\mathcal{M}_y$, respectively. Next, we give a linear-time and linear-space solution for the MAW-SequenceComparison problem. It is known from [9] and [2] that we can compute the sets $\mathcal{M}_x$ and $\mathcal{M}_y$ in linear time and space with respect to the two lengths m and n, respectively. The idea of our strategy consists of a merge sort on the sets $\mathcal{M}_x$ and $\mathcal{M}_y$, after they have been ordered with the help of suffix arrays.

To this end, we construct the suffix array associated to the word $w = xy$, together with the implicit LCP array corresponding to it. All of these structures can be constructed in time and space $\mathcal{O}(m+n)$, as mentioned earlier. Furthermore, we can preprocess the array LCP for range minimum queries, which we denote by $\mathsf{RMQ}_{\mathsf{LCP}}$ [13]. With the preprocessing complete, the longest common prefix LCE of two suffixes of w starting at positions p and q can be computed in constant time [19], using the formula $\mathsf{LCE}(w,p,q) = \mathsf{LCP}[\mathsf{RMQ}_{\mathsf{LCP}}(\mathsf{iSA}[p]+1, \mathsf{iSA}[q])]$.

Using these data structures, it is straightforward to sort the tuples in the sets $\mathcal{M}_x$ and $\mathcal{M}_y$ lexicographically. That is, two tuples $x_1, x_2 \in \mathcal{M}_x$, are ordered according to the letter following their longest common prefix, or when it is not the case, with the one being the prefix, coming first. To do this, we simply go once through the suffix array associated to w and assign to each tuple in $\mathcal{M}_x$, respectively $\mathcal{M}_y$, the rank of the suffix starting at the position indicated by its second component, in the suffix array. Since sorting an array of n distinct integers, such that each is in $[0, n-1]$, can be done in time $\mathcal{O}(n)$ (using bucket sort, for example), we can sort now each of the sets of minimal absent words, taking into consideration the letter on the first position and these ranks. Thus, from now on, we assume that $\mathcal{M}_x = (x_0, x_1, \ldots, x_k)$ where x_i is lexicographically smaller than x_{i+1}, for $0 \leq i < k \leq \sigma m$, and $\mathcal{M}_y = (y_0, y_1, \ldots, y_\ell)$, where y_j is lexicographically smaller than y_{j+1}, for $0 \leq j < \ell \leq \sigma n$.

Provided these tools, we now proceed to do the merge. Thus, considering that we are analysing the $(i+1)$th tuple in $\mathcal{M}_x$ and the $(j+1)$th tuple in $\mathcal{M}_y$, we note that the two are equal if and only if $x_i[0] = y_j[0]$ and

$$\mathsf{LCE}(w, x_i[1], |x| + y_j[1]) \geq \ell, \text{ where } \ell = x_i[2] - x_i[1] = y_j[2] - y_j[1].$$

In other words, the two minimal absent words are equal if and only if their first letters coincide, they have equal length $\ell + 1$, and the longest common prefix of

the suffixes of w starting at the positions indicated by the second components of the tuples has length at least ℓ.

Such a strategy will empower us with the means for constructing a new set $\mathcal{M}_{xy} = \mathcal{M}_x \cup \mathcal{M}_y$. At each step, when analysing tuples x_i and y_j we proceed as following:

$$\mathcal{M}_{xy} = \begin{cases} \mathcal{M}_{xy} \cup \{x_i\}, & \text{and increment } i, & \text{if } x_i < y_j; \\ \mathcal{M}_{xy} \cup \{y_j\}, & \text{and increment } j, & \text{if } x_i > y_j; \\ \mathcal{M}_{xy} \cup \{x_i = y_j\}, & \text{and increment both } i \text{ and } j, & \text{if } x_i = y_j. \end{cases}$$

Observe that the last condition is saying that basically each common tuple is added only once to their union.

Furthermore, simultaneously with this construction we can also calculate the similarity between the words, given by $\mathsf{LW}(\mathcal{M}_x, \mathcal{M}_y)$, which is initially set to 0. Thus, at each step, when comparing the tuples x_i and y_j, we update

$$\mathsf{LW}(\mathcal{M}_x, \mathcal{M}_y) = \begin{cases} \mathsf{LW}(\mathcal{M}_x, \mathcal{M}_y) + \frac{1}{|x_i|^2}, & \text{and increment } i, & \text{if } x_i < y_j; \\ \mathsf{LW}(\mathcal{M}_x, \mathcal{M}_y) + \frac{1}{|y_j|^2}, & \text{and increment } j, & \text{if } x_i > y_j; \\ \mathsf{LW}(\mathcal{M}_x, \mathcal{M}_y), & \text{and increment both } i \text{ and } j, & \text{if } x_i = y_j. \end{cases}$$

We impose the increment of both i and j in the case of equality as in this case we only look at the symmetric difference between the sets of minimal absent words.

As all these operations take constant time, once per each tuple in $\mathcal{M}_x$ and $\mathcal{M}_y$, it is easily concluded that the whole operation takes in the case of a fixed-sized alphabet time and space $\mathcal{O}(m+n)$. Thus, we can compute the symmetric difference between the *complete* sets of minimal absent words, as opposed to [7], of two words defined over a fixed-sized alphabet, in linear time and space with respect to the lengths of the two words. We obtain the following result.

Theorem 1. *Problem* MAW-SequenceComparison *can be solved in time and space* $\mathcal{O}(m+n)$.

4 Circular Sequence Comparison

Next, we discuss two possible definitions for the minimal absent words of a circular word, and highlight the differences between them.

We start by recalling some basic facts about minimal absent words. For further details and references the reader is recommended [11]. Every factorial language L is uniquely determined by its (antifactorial) language of minimal absent words $\mathcal{M}_L$, through the equation $L = \Sigma^* \setminus \Sigma^* \mathcal{M}_L \Sigma^*$. The converse is also true, since by the definition of a minimal absent word we have $\mathcal{M}_L = \Sigma L \cap L\Sigma \cap (\Sigma^* \setminus L)$. The previous equations define a bijection between factorial and antifactorial languages. Moreover, this bijection preserves regularity. In the case of a single (linear) word x, the set of minimal absent words for x is indeed the antifactorial language $\mathcal{M}_{\mathcal{F}_x}$. Furthermore, we can retrieve x from its set of minimal absent words in linear time and space [9].

Recall that given a circular word $\tilde{x}$, the set $\mathcal{F}_{\tilde{x}}$ of factors of $\tilde{x}$ is equal to the set $\mathcal{F}_{xx} \cap \Sigma^{\leq |x|}$ of factors of xx whose lengths are at most $|x|$, where x is any linearization of $\tilde{x}$. Since a circular word $\tilde{x}$ is a conjugacy class containing all the rotations of a linear word x, the language $\mathcal{F}_{\tilde{x}}$ can be seen as the factorial closure of the set $\{x^{\langle i \rangle} \mid i = 0, \ldots, |x| - 1\}$. This leads to the first definition of the set of minimal absent words for $\tilde{x}$, that is the set $\mathcal{M}_{\mathcal{F}_{\tilde{x}}} = \{aub \mid a, b \in \Sigma, aub \notin \mathcal{F}_{\tilde{x}}, au, ub \in \mathcal{F}_{\tilde{x}}\}$. For instance, if $x = abaab$, we have

$$\mathcal{M}_{\mathcal{F}_{\tilde{x}}} = \{aaa, aabaa, aababa, abaaba, ababaa, baabab, babaab, babab, bb\}.$$

The advantage of this definition is that we can retrieve uniquely $\tilde{x}$ from $\mathcal{M}_{\mathcal{F}_{\tilde{x}}}$. However, the total size of $\mathcal{M}_{\mathcal{F}_{\tilde{x}}}$ (that is, the sum of the lengths of its elements) can be very large, as the following lemma suggests.

Lemma 2. *Let $\tilde{x}$ be a circular word of length $m > 0$. The set $\mathcal{M}_{\mathcal{F}_{\tilde{x}}}$ contains precisely ℓ words of maximal length $m+1$, where ℓ is the number of distinct rotations of any linearization x of $\tilde{x}$, that is, the cardinality of $\{x^{\langle i \rangle} \mid i = 0, \ldots, |x| - 1\}$.*

Proof. Let $x = x[0]x[1]\ldots x[m-1]$ be a linearization of $\tilde{x}$. The word obtained by appending to x its first letter, $x[0]x[1]\ldots x[m-1]x[0]$, belongs to $\mathcal{M}_{\mathcal{F}_{\tilde{x}}}$, since it has length $m+1$, hence it cannot belong to $\mathcal{F}_{\tilde{x}}$, but its maximal proper prefix $x = x^{\langle 0 \rangle}$ and its maximal proper suffix $x^{\langle 1 \rangle} = x[1]\ldots x[m-1]x[0]$ belong to $\mathcal{F}_{\tilde{x}}$.

The same argument shows that for any rotation $x^{\langle i \rangle} = x[i]x[i+1]\ldots x[m-1]x[0]\ldots x[i-1]$ of x, the word $x[i]x[i+1]\ldots x[m-1]x[0]\ldots x[i-1]x[i]$, obtained by appending to $x^{\langle i \rangle}$ its first letter, belongs to $\mathcal{M}_{\mathcal{F}_{\tilde{x}}}$.

Conversely, if a word of maximal length $m+1$ is in $\mathcal{M}_{\mathcal{F}_{\tilde{x}}}$, then its maximal proper prefix and its maximal proper suffix are words of length m in $\mathcal{F}_{\tilde{x}}$, so they must be consecutive rotations of x.

Therefore, the number of words of maximal length $m+1$ in $\mathcal{M}_{\mathcal{F}_{\tilde{x}}}$ equals the number of distinct rotations of x, hence the statement follows. □

This is in sharp contrast with the situation for linear words, where the set of minimal absent words can be represented on a trie having size linear in the length of the word. Indeed, the algorithm MF-TRIE, introduced in [9], builds the tree-like deterministic automaton accepting the set of minimal absent words for a word x taking as input the factor automaton of x, that is the minimal deterministic automaton recognizing the set of factors of x. The leaves of the trie correspond to the minimal absent words for x, while the internal states are those of the factor automaton. Since the factor automaton of a word x has less than $2|x|$ states (for details, see [8]), this provides a representation of the minimal absent words of a word of length n in space $O(\sigma n)$.

This algorithmic drawback leads us to the second definition. This second definition of minimal absent words for circular strings has been already introduced in [27,26]. First, we give a combinatorial result which shows that when considering circular words it does not make sense to look at absent words obtained from more than two rotations.

Lemma 3. *For any positive integer k and any word u, the set $V = \{v \mid k|u|+1 < |v| \leq (k+1)|u|\} \cap (\mathcal{M}_{u^{k+1}} \setminus \mathcal{M}_{u^k})$ is empty.*

Proof. This obviously holds for all words u of length 1. Assume towards a contradiction that this is not the case in general. Hence, there must exist a word v of length m that fulfills the conditions in the lemma, thus $v \in V$ and $m > 2$. Furthermore, since the length $m-1$ prefix and the length $m-1$ suffix of every minimal absent word occur in the main word at non-consecutive positions, there must exist positions $i < j \leq n = |u|$ such that

$$v[1\,..\,m-2] = u^{k+1}[i+1\,..\,i+m-2] = u^{k+1}[j+1\,..\,j+m-2]. \tag{1}$$

Obviously, following Equation (1), since $m-2 \geq kn$, we have that $v[1\,..\,m-2]$ is $(j-i)$-periodic. But, we know that $v[1\,..\,m-2]$ is also n-periodic. Thus, following a direct application of the periodicity lemma we have that $v[1\,..\,m-2]$ is $p = \gcd(j-i, n)$-periodic. But, in this case we have that u is p-periodic, and, therefore, $u[i] = u[j]$, which leads to a contradiction with the fact that v is a minimal absent word, whenever i is defined. Thus, it must be the case that $i = -1$. Using the same strategy and looking at positions $u[i+m-2]$ and $u[j+m-2]$, we conclude that $j+m-2 = (k+1)n$. Therefore, in this case, we have that $m = kn+1$, which is a contradiction with the fact that the word v fulfills the conditions of the lemma. This concludes the proof. □

Observe now that the set V consists in fact of all extra minimal absent words generated whenever we look at more than one rotation, that do not include the length arguments. That is, V does not include the words bounding the maximum length that a word is allowed, nor the words created, or lost, during a further concatenation of an image of u. However, when considering an iterative concatenation of the word, these extra elements determined by the length constrain cancel each other.

As observed in Section 2, two rotations of the same word x generate two languages that have the same set of factors. So, we can unambiguously associate to a circular word $\tilde{x}$ the (infinite) factorial language $\mathcal{F}_{\tilde{x}^*}$. It is therefore natural to define the set of minimal absent words for the circular word $\tilde{x}$ as the set $\mathcal{M}_{\mathcal{F}_{\tilde{x}^*}}$. For instance, if $\tilde{x} = abaab$, then we have

$$\mathcal{M}_{\mathcal{F}_{\tilde{x}^*}} = \{aaa, aabaa, babab, bb\}.$$

This second definition is much more efficient in terms of space, as we show below. In particular, the length of the words in $\mathcal{M}_{\mathcal{F}_{\tilde{x}^*}}$ is bounded from above by $|x|$, hence $\mathcal{M}_{\mathcal{F}_{\tilde{x}^*}}$ is a finite set.

Recall that a word x is *a power* of a word y if there exists a positive integer $k > 1$ such that x is expressed as k consecutive concatenations of y, denoted by $x = y^k$. Conversely, a word x is *primitive* if $x = y^k$ implies $k = 1$. Notice that a word is primitive if and only if any of its rotation is. We can therefore extend the definition of primitivity to circular words. The definition of $\mathcal{M}_{\mathcal{F}_{\tilde{x}^*}}$ does not allow one to uniquely reconstruct $\tilde{x}$ from $\mathcal{M}_{\mathcal{F}_{\tilde{x}^*}}$, unless $\tilde{x}$ is known

to be primitive, since it is readily verified that $\mathcal{F}_{\tilde{x}^*} = \mathcal{F}_{\widetilde{x\tilde{x}}^*}$ and therefore also the minimal absent words of these two languages coincide. However, from the algorithmic point of view, this issue can be easily managed by storing the length $|x|$ of a linearization x of $\tilde{x}$ together with the set $\mathcal{M}_{\mathcal{F}_{\tilde{x}^*}}$. Moreover, in most practical cases, for example when dealing with biological sequences, it is highly unlikely that the circular word considered is not primitive.

The difference between the two definitions above is presented in the next lemma.

Lemma 4. $\mathcal{M}_{\mathcal{F}_{\tilde{x}^*}} = \mathcal{M}_{\mathcal{F}_{\tilde{x}}} \cap \Sigma^{\leq |x|}$.

Proof. Clearly, $\mathcal{F}_{\tilde{x}^*} \cap \Sigma^{\leq |x|} = \mathcal{F}_{\tilde{x}}$. The statement then follows from the definition of minimal absent words. □

Based on the previous discussion, we set $\mathcal{M}_{\tilde{x}} = \mathcal{M}_{\mathcal{F}_{\tilde{x}^*}}$, while the following corollary comes straightforwardly as a consequence of Lemma 3.

Corollary 5. *Let $\tilde{x}$ be a circular word. Then $\mathcal{M}_{\tilde{x}} = \mathcal{M}_{xx}^{|x|}$.*

Corollary 5 was first introduced as a definition for the set of minimal absent words of a circular word in [26]. Using the result of Corollary 5, we can easily extend the algorithm described in the previous section to the case of circular words. That is, given two circular words $\tilde{x}$ of length m and $\tilde{y}$ of length n, we can compute in time and space $\mathcal{O}(m+n)$ the quantity $\mathsf{LW}(\mathcal{M}_{\tilde{x}}, \mathcal{M}_{\tilde{y}})$. We obtain the following result.

Theorem 6. *Problem* MAW-CircularSequenceComparison *can be solved in time and space $\mathcal{O}(m+n)$.*

5 Implementation and Applications

We implemented the presented algorithms as programme scMAW to perform pairwise sequence comparison for a set of sequences using minimal absent words. scMAW uses programme MAW [2] for linear-time and linear-space computation of minimal absent words using suffix array. scMAW was implemented in the C programming language and developed under GNU/Linux operating system. It takes, as input argument, a file in MultiFASTA format with the input sequences, and then any of the two methods, for *linear* or *circular* sequence comparison, can be applied. It then produces a file in PHYLIP format with the distance matrix as output. Cell $[x, y]$ of the matrix stores $\mathsf{LW}(\mathcal{M}_x, \mathcal{M}_y)$ (or $\mathsf{LW}(\mathcal{M}_{\tilde{x}}, \mathcal{M}_{\tilde{y}})$ for the circular case). The implementation is distributed under the GNU General Public License (GPL), and it is available at `http://github.com/solonas13/maw`, which is set up for maintaining the source code and the man-page documentation. Notice that *all* input datasets and the produced outputs referred to in this section are publicly maintained at the same web-site.

An important feature of the proposed algorithms is that they require space linear in the length of the sequences (see Theorem 1 and Theorem 6). Hence, we

were also able to implement scMAW using the Open Multi-Processing (OpenMP) PI for shared memory multiprocessing programming to distribute the workload across the available processing threads without a large memory footprint.

Application. Recently, there has been a number of studies on the biological significance of absent words in various species [1,16,31]. In [16], the authors presented dendrograms from dinucleotide relative abundances in sets of minimal absent words for prokaryotes and eukaryotic genomes. The analyses support the hypothesis that minimal absent words are inherited through a common ancestor, in addition to lineage-specific inheritance, only in vertebrates. Very recently, in [31], it was shown that there exist three minimal words in the Ebola virus genomes which are absent from human genome. The authors suggest that the identification of such species-specific sequences may prove to be useful for the development of both diagnosis and therapeutics.

In this section, we show a potential application of our results for the construction of dendrograms for DNA sequences with circular structure. Circular DNA sequences can be found in viruses, as plasmids in archaea and bacteria, and in the mitochondria and plastids of eukaryotic cells. Circular sequence comparison thus finds applications in several contexts such as reconstructing phylogenies using viroids RNA [24] or Mitochondrial DNA (MtDNA) [17]. Conventional tools to align circular sequences could yield an incorrectly high genetic distance between closely-related species. Indeed, when sequencing molecules, the position where a circular sequence starts can be totally arbitrary. Due to this *arbitrariness*, a suitable rotation of one sequence would give much better results for a pairwise alignment [4,18]. In what follows, we demonstrate the power of minimal absent words to pave a path to resolve this issue by applying Corollary 5 and Theorem 6. Next we do not claim that a solid phylogenetic analysis is presented but rather an investigation for potential applications of our theoretical findings.

We performed the following experiment with synthetic data. First, we simulated a basic dataset of DNA sequences using INDELible [14]. The number of taxa, denoted by α, was set to 12; the length of the sequence generated at the root of the tree, denoted by β, was set to 2500bp; and the substitution rate, denoted by γ, was set to 0.05. We also used the following parameters: a deletion rate, denoted by δ, of 0.06 *relative* to substitution rate of 1; and an insertion rate, denoted by ϵ, of 0.04 *relative* to substitution rate of 1. The parameters were chosen based on the genetic diversity standard measures observed for sets of MtDNA sequences from primates and mammals [4]. We generated another instance of the basic dataset, containing one *arbitrary* rotation of each of the α sequences from the basic dataset. We then used this randomized dataset as input to scMAW by considering $\mathsf{LW}(\mathcal{M}_{\tilde{x}}, \mathcal{M}_{\tilde{y}})$ as the distance metric. The output of scMAW was passed as input to NINJA [33], an efficient implementation of neighbor-joining [30], a well-established hierarchical clustering algorithm for inferring dendrograms (trees). We thus used NINJA to infer the respective tree T_1 under the neighbor-joining criterion. We also inferred the tree T_2 by following the same pipeline, but by considering $\mathsf{LW}(\mathcal{M}_x, \mathcal{M}_y)$ as distance metric, as well as the tree T_3 by using the *basic* dataset as input of this pipeline and $\mathsf{LW}(\mathcal{M}_{\tilde{x}}, \mathcal{M}_{\tilde{y}})$

Dataset $< \alpha, \beta, \gamma, \delta, \epsilon >$	T_1 vs. T_3	T_2 vs. T_3
$< 12, 2500, 0.05, 0.06, 0.04 >$	100%	100%
$< 12, 2500, 0.20, 0.06, 0.04 >$	100%	88,88%
$< 12, 2500, 0.35, 0.06, 0.04 >$	100%	100%
$< 25, 2500, 0.05, 0.06, 0.04 >$	100%	100%
$< 25, 2500, 0.20, 0.06, 0.04 >$	100%	100%
$< 25, 2500, 0.35, 0.06, 0.04 >$	100%	100%
$< 50, 2500, 0.05, 0.06, 0.04 >$	100%	97,87%
$< 50, 2500, 0.20, 0.06, 0.04 >$	100%	97,87%
$< 50, 2500, 0.35, 0.06, 0.04 >$	100%	100%

Table 1. Accuracy measurements based on relative pairwise RF distance

as distance metric. Hence, notice that T_3 represents the original tree. Finally, we computed the pairwise Robinson-Foulds (RF) distance [29] between: T_1 and T_3; and T_2 and T_3.

Let us define *accuracy* as the difference between 1 and the relative pairwise RF distance. We repeated this experiment by simulating different datasets $< \alpha, \beta, \gamma, \delta, \epsilon >$ and measured the corresponding accuracy. The results in Table 1 (see T_1 vs. T_3) suggest that by considering $\mathsf{LW}(\mathcal{M}_{\tilde{x}}, \mathcal{M}_{\tilde{y}})$ we can always reconstruct the original tree even if the sequences have first been arbitrarily rotated (Corollary 5). This is not the case (see T_2 vs. T_3) if we consider $\mathsf{LW}(\mathcal{M}_x, \mathcal{M}_y)$. Notice that 100% accuracy denotes a (relative) pairwise RF distance of 0.

6 Final Remarks

In this article, complementary to measures that refer to the composition of sequences in terms of their constituent patterns, we considered sequence comparison using minimal absent words, information about what does not occur in the sequences. We presented the first linear-time and linear-space algorithm to compare two sequences by considering *all* their minimal absent words (Theorem 1). In the process, we presented some results of combinatorial interest, and also extended the proposed techniques to circular sequences. The power of minimal absent words is highlighted by the fact that they provide a tool for sequence comparison that is as efficient for circular as it is for linear sequences (Corollary 5 and Theorem 6); whereas, this is not the case, for instance, using the general edit distance model [21]. Finally, a preliminary experimental study shows the potential of our theoretical findings.

Our immediate target is to consider the following *incremental* version of the same problem: given an appropriate encoding of a comparison between sequences x and y, can one incrementally compute the answer for x and ay, and the answer for x and ya, efficiently, where a is an additional letter? Incremental sequence comparison, under the edit distance model, has already been considered in [20].

In [18], the authors considered a more powerful generalization of the q-gram distance (see [32] for definition) to compare x and y. This generalization comprises partitioning x and y in β blocks each, as evenly as possible, computing the q-gram distance between the corresponding block pairs, and then summing

up the distances computed blockwise to obtain the new measure. We are also planning to apply this generalization to the similarity measure studied here and evaluate it using real and synthetic data.

Acknowledgements

We warmly thank Alice Heliou for her inestimable code contribution and Antonio Restivo for useful discussions. Gabriele Fici's work was supported by the PRIN 2010/2011 project "Automi e Linguaggi Formali: Aspetti Matematici e Applicativi" of the Italian Ministry of Education (MIUR) and by the "National Group for Algebraic and Geometric Structures, and their Applications" (GNSAGA – INdAM). Robert Mercaş's work was supported by the P.R.I.M.E. programme of DAAD co-funded by BMBF and EU's 7th Framework Programme (grant 605728). Solon P. Pissis's work was supported by a Research Grant (#RG130720) awarded by the Royal Society.

References

1. Acquisti, C., Poste, G., Curtiss, D., Kumar, S.: Nullomers: Really a matter of natural selection? PLoS ONE 2(10) (2007)
2. Barton, C., Heliou, A., Mouchard, L., Pissis, S.P.: Linear-time computation of minimal absent words using suffix array. BMC Bioinformatics 15, 388 (2014)
3. Barton, C., Heliou, A., Mouchard, L., Pissis, S.P.: Parallelising the computation of minimal absent words. In: PPAM. LNCS (2015)
4. Barton, C., Iliopoulos, C.S., Kundu, R., Pissis, S.P., Retha, A., Vayani, F.: Accurate and efficient methods to improve multiple circular sequence alignment. In: SEA, LNCS, vol. 9125, pp. 247–258 (2015)
5. Béal, M., Mignosi, F., Restivo, A., Sciortino, M.: Forbidden words in symbolic dynamics. Advances in Applied Mathematics 25(2), 163–193 (2000)
6. Belazzougui, D., Cunial, F., Kärkkäinen, J., Mäkinen, V.: Versatile succinct representations of the bidirectional Burrows–Wheeler transform. In: ESA, LNCS, vol. 8125, pp. 133–144 (2013)
7. Chairungsee, S., Crochemore, M.: Using minimal absent words to build phylogeny. Theoretical Computer Science 450(0), 109–116 (2012)
8. Crochemore, M., Hancart, C., Lecroq, T.: Algorithms on Strings. Cambridge University Press, New York, NY, USA (2007)
9. Crochemore, M., Mignosi, F., Restivo, A.: Automata and forbidden words. Information Processing Letters 67, 111–117 (1998)
10. Domazet-Lošo, M., Haubold, B.: Efficient estimation of pairwise distances between genomes. Bioinformatics 25(24), 3221–3227 (2009)
11. Fici, G.: Minimal Forbidden Words and Applications. Ph.D. thesis, Université de Marne-la-Vallée (2006)
12. Fischer, J.: Inducing the LCP-array. In: WADS, LNCS, vol. 6844, pp. 374–385 (2011)
13. Fischer, J., Heun, V.: Space-efficient preprocessing schemes for range minimum queries on static arrays. SIAM Journal of Computing 40(2), 465–492 (2011)

14. Fletcher, W., Yang, Z.: INDELible: A flexible simulator of biological sequence evolution. Molecular Biology and Evolution 26(8), 1879–1888 (2009)
15. Fukae, H., Ota, T., Morita, H.: On fast and memory-efficient construction of an antidictionary array. In: ISIT. pp. 1092–1096. IEEE (2012)
16. Garcia, S.P., Pinho, O.J., Rodrigues, J.M.O.S., Bastos, C.A.C., G, P.J.S.: Minimal absent words in prokaryotic and eukaryotic genomes. PLoS ONE 6 (2011)
17. Goios, A., Pereira, L., Bogue, M., Macaulay, V., Amorim, A.: mtDNA phylogeny and evolution of laboratory mouse strains. Genome Research 17(3), 293–298 (2007)
18. Grossi, R., Iliopoulos, C.S., Mercaş, R., Pisanti, N., Pissis, S.P., Retha, A., Vayani, F.: Circular sequence comparison with q-grams. In: WABI, LNCS, vol. 9289, pp. 203–216 (2015)
19. Ilie, L., Navarro, G., Tinta, L.: The longest common extension problem revisited and applications to approximate string searching. Journal of Discrete Algorithms 8(4), 418–428 (2010)
20. Landau, G.M., Myers, E.W., Schmidt, J.P.: Incremental string comparison. SIAM J. Comput. 27(2), 557–582 (1998)
21. Maes, M.: On a cyclic string-to-string correction problem. Information Processing Letters 35(2), 73–78 (1990)
22. Manber, U., Myers, E.W.: Suffix arrays: A new method for on-line string searches. SIAM Journal of Computing 22(5), 935–948 (1993)
23. Mignosi, F., Restivo, A., Sciortino, M.: Words and forbidden factors. Theoretical Computer Science 273(1-2), 99–117 (2002)
24. Mosig, A., Hofacker, I.L., Stadler, P.F.: Comparative Analysis of Cyclic Sequences: Viroids and other Small Circular RNAs. In: GCB, LNI, vol. 83, pp. 93–102 (2006)
25. Nong, G., Zhang, S., Chan, W.H.: Linear suffix array construction by almost pure induced-sorting. In: DCC. pp. 193–202. IEEE (2009)
26. Ota, T., Morita, H.: On a universal antidictionary coding for stationary ergodic sources with finite alphabet. In: ISITA. pp. 294–298. IEEE (2014)
27. Ota, T., Morita, H.: On antidictionary coding based on compacted substring automaton. In: ISIT. pp. 1754–1758. IEEE (2013)
28. Pinho, A.J., Ferreira, P.J.S.G., Garcia, S.P.: On finding minimal absent words. BMC Bioinformatics 11 (2009)
29. Robinson, D., Fould, L.: Comparison of phylogenetic trees. Mathematical Biosciences 53(1-2), 131–147 (1981)
30. Saitou, N., Nei, M.: The neighbor-joining method: a new method for reconstructing phylogenetic trees. Molecular Biology and Evolution 4(4), 406–425 (1987)
31. Silva, R.M., Pratas, D., Castro, L., Pinho, A.J., Ferreira, P.J.S.G.: Three minimal sequences found in Ebola virus genomes and absent from human DNA. Bioinformatics (2015)
32. Ukkonen, E.: Approximate string-matching with q-grams and maximal matches. Theoretical Computer Science 92(1), 191–211 (1992)
33. Wheeler, T.J.: Large-scale neighbor-joining with NINJA. In: WABI, LNCS, vol. 5724, pp. 375–389 (2009)